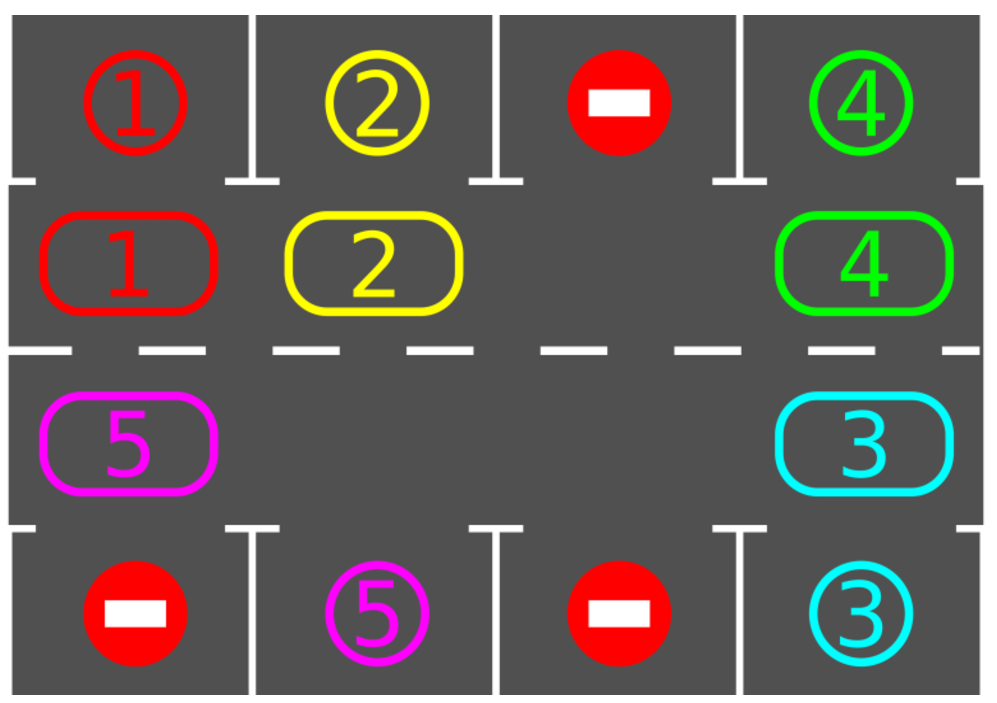
1. Codeforces round 492 div1 A:

题意：给一个4 \* N（N<=50） 的停车场， 第一行和第四行可以停车，你有k<=2\*N 辆车在2&3行，车从1-k编号，你需把每个编号i的车停到停车位置i。求一个两万步以内的解决方案，如果没有，输出-1.注意车不可以在第一行和第四行移动。



思路：考虑怎么构造答案，首先很显然我们把任意一辆车直接走一圈一定可以找到车位，所以考虑每辆车O2\*n的寻找车位，然后对于k辆车，我们的解的集合大小是O2\*N\*K = 100\*100 < 20000，这样我们就构造出了一组答案。那么无解的情况显然，当没有一辆车可以移动的时候就无解，这只有可能出现在k=2\*N且无车可以直接停进去的时候，否则我们可以通过拖动车的队列来不断寻找相遇。

时间复杂度：O2\*K\*N;

思维难度：简单。

代码难度：100行左右，熟悉位运算和stl可以压到30行，但是难以调试，不建议。

综合难度: easy+;

1. Codeforces round 492 div1 F:

题意：给一个有根树，1是根，树最多有3000个点，现在每个点有一个点权，规定：一个点的权值不可以大于他的任意一级祖先的权值，现在给定D，意义是这n个点的权值都在闭区间[1, D] 内， D <= 1e9. 问：有多少种分配点权的方式，模意义输出。

思路：如果D的范围开个根号，这个题用ON\*D的动态规划，即f(i, j) 表示根节点是i，且权值是j有几种方案，这样对于他的每个儿子，运用加法原理可以得到他对父亲的选择方案的贡献，对于一个节点的所有儿子，他可能的取值就是把所有的儿子的贡献值运用乘法原理计算。状态转移是显然的，假设父亲要选i，儿子满足他的值是所有的小于等于i，那么前缀和统计转移即可。

好了，现在有了这个签到dp的铺垫，我们考虑开大D，那么我们观察这样的多项式。

F = a1 \* x^1 + a2 \* x^2 + … +aD \* x^D, ^代表求幂运算。那么我们实际上需要计算的就是Σai, i<=D, 如果ai表示幂次是i的选择方案，显然这个Σ就是我们要的答案。

那么同样，使用上面的dp，我们很容易可以算出这个多项式的前N项，对于val <= D， dp求得我们要的结果，那么D那么大，我们可以用拉格朗日插值，ON的处理出D项的系数和。

时间复杂度：O N \* N + N

代码复杂度：拉格朗日插值模板。

思维复杂度：高，要联想到多项式解决问题。

难度：hard。

1. Codeforces edu 46 E：

给一张无向图，你可以选择任意节点s作为开始节点，任意e作为结束点，对于这两个点的简单路径上的所有边，假设我们摧毁这条边之后能把图拆成两半，你就可以在这个位置放上妖怪，问任意选择s, t的情况下，我们最多可以放多少妖怪。

思路：不是桥的数目，首先必须是存在于s<->t的简单路径上的边才可以，所以求桥的也就跑过样例了。

但是和桥有关系，首先妖怪必须放在桥上，那么对于桥树（每一条边都是原始图的一个桥）而言，我们只要找这棵树的直径(树上最远点对)就做完了。

时间复杂度: O(m + n).

思维复杂度：中等。

代码难度：低。

难度：mid-；

1. 一棵树，每条边有个正向边权有个反向边权，然后Q次询问，每次问两个点，让你从u->v，每条路径最多经过两遍，问路径最大权值和。

思路：容斥+LCA灌水

时间复杂度O(n + Q \* log(n))

思维复杂度：easy

代码复杂度：mid-

难度：easy+

1. 一张图最开始没有边，只有至多5000个点，考虑下面五十万个操作，操作1：加入一条本不存在的边，操作2：删除一条本存在的边，操作3：查询一条边现在是否存在。

思路：首先这个题没有强制在线，那么离线处理。

我们把这个问题拆成两个问题，确定一条边的存在时间，这个可以用map+线段树套vector比较容易的解决。第二个问题：假设这个问题有询问那么多个时间点，对于一个区间，我们知道有一些边存在，那么如果出现一个区间不完全被这个区间包含，我们需要取消这个区间对答案的影响，那么，可持久化并查集恰好满足要求（回滚，检查是否联通，合并），然后就是个从上面爆搜下来的操作，当我们退出一段区间的时候，把这个区间内所有影响（这个区间新增加的联通关系）全部回滚即可。

时间复杂度:O(M \* logM \* logM)

思维复杂度：mid

代码复杂度：mid+

综合难度：mid+

6 :

（1）.一段区间四个操作, [l, r] + c [l, r] / d query->min( [l, r] ), query->sum( [l, r] )

N, Q <= 1e5, 2 <= d <= 2e9, abs(c) <= 1e4

（2）.上面的除法操作改成乘法，同时对1e9+7取模。

这两个题专治线段树只会抄板子的。

首先第二个乘法问题相对比较简单，我们可以来考虑乘法问题作为基础，在这之前，如果对线段树lazy tag不熟悉，建议先去熟悉一下。对于乘法和加法的组合问题，我们要人为的给lazy标记优先级，也就是说，观察 ((val + x1) \* y1 + x2 + x3) \* y2 \* y3这类运算，我们发现加法并不影响前面的乘法，而乘法对前面的所有的加法都会差生影响，那么我们定义两种标记：加乘标记和乘法标记，对于加乘标记，定义为经过一系列运算之后的加法tag，对于例子给的式子，就是x1 -> x1 \* y1 - > x1 \* y1 + x2 + x3->(x1 \* y1 + x2 +x3)可以总结到，加乘标记就是按照操作的时间顺序来搞，然后把这个加乘标记直接加到原区间即可，乘法标记就是平时常用的那种，直接推下去即可。清楚一点的表示就是，上面的式子经过最简单的展开，可以化简为 y1\*y2\*y3\*val + (x1\*y1+x2+x3)\*y2\*y3, 观察得知，前面的就是乘法标记，后面的就是加乘标记，我们下推标记之后，实际上就是val\*乘法标记+加法标记，就变成了互相独立的两个标记，val = val \* tag1 + tag2即可。标记的选择与定义非常的自然，这是根据下推的数学关系推出的。

那么回来看第一问，第一问也是两个标记，但是实际上我们可以取巧打一个标记，就是加法标记，除法可以看成特殊的减法，什么时候可以减呢, 当考虑对区间最大值进行除法运算和对区间最小值除法运算之后发现运算后的值和之前的值的差相等，就是可以的。但是这个复杂度是个玄学式子，和初始序列有多混乱有关，也和后面的加法操作的区间的交集的个数和每次加法可以贡献的混乱度有关，反正算不出来，最大单次nlog, 期望复杂度是log次nlog的查询之后变成O1查询。但是这玩意太玄学了，我们重新定义加乘操作就做完了,重新写一下那个运算式，就会发现val / (y1\*y2\*y3) + x1/(y1\*y2\*y3) + x2/(y2\*y3)这个样子，所以我们每次进来一组除法，就把它和之前的“加除”标记比一下，再和除法标记乘一下，就做完了。

大概就是介绍一下push\_down的技巧，

难度：

中等

比较无脑难写。

7

题意：给一个5\*10^5长度的序列，你需要执行两种操作，第一种给一个区间和值K，你需要把区间所有的值和k取max， 第二个操作是查询，给出区间和值K, 值X，你需要回答在区间内比k严格小的x个数依次是多少，所有询问的x加起来不超过5\*10^6,每一次询问的x不超过10^5.

思路：这个题先想一下如何确认x个最小值，确认一个：线段树区间查询最小值，那么我们可以维护最小值和最小值的位置，然后我们就把大区间分成了两个，然后我们分别在两边查询？不对。。你要在小的那边查询出第二小值，第三小值可能也在小的那里，所以绑起来丢到堆里面，这样处理，每次询问至多x次处理，每次只有查询+堆插入，是log的，所以总复杂度是n \* log + Σx \* log， 需要线段树维护区间最小值和他的下标，需要堆来维护当前能拿出来找下一个值的区间[l, r], 假设我们找到这段区间的最小值在pos, 我们就把[l, pos - 1] 和[pos + 1, r]这两个里面的合法区间丢进堆里。

思维难度：hard-（也可能是我太困了）

代码难度：mid+（其实把一个小写l打成了数字1查了半个小时）。

难度：mid+

8:

题意：给你一个3e5长度的01序列，你有两种消费不同的操作，翻转一个字串和反转一个字串，反转就是0->1,1->0， 问花多少钱能把他变成都是1.

思路：我们至少要有一次反转操作，这种情况下，注意到翻转会让0的块数-1，所以他的花费是C(反转) + （x - 1）\* C(翻转)， 假设x是0的块数。另一种情况是x \* 反转，两个取min就行了，因为我们注意到这两个其实是 C(反转) + （x-1）\*C(反转/翻转)，所以哪个小选哪个。

时间复杂度：On

思维复杂度：mid-

代码复杂度：easy

综合复杂度：easy+

9：

给你n个四种罗马数字，问你能表示出几个数字，四种罗马数字代表1, 2, 5, 10，然后若干个罗马数字拼起来就是让他们相加。N <= 1e9

有个很显然的4^nlog的大暴力（划掉）

这个显然的暴力只能用来打表。。。

赛场上：打出这个表之后丢进拉格朗日插值计算找规律，发现在15项以后数列的通项向fn = 2 \* fn-1 – fn-2 稳定，后面都是无穷小项，这个关系是显然的等差数列，然后就赢了，20以下暴力打表，20以上直接计算(f(21) – f(20)) \* (n - 20) + f(20)即可;

时间复杂度：4^20log

思维复杂度：暴力没思维复杂度，最开始没想到打表，想dp，但是没想出来，后来打了表看出规律登向rk1巅峰（不是）。

代码复杂度：easy

综合复杂度：mid

10，11，12，13，14，15，16： 7.7；

17，18，19： 7.7codeM